Input/Output 62

In addition to transferring data to and from external memory, Intel Architecture processors can also transfer data to and from input/output ports (I/O ports). I/O ports are created in system hardware by circuity that decodes the control, data, and address pins on the processor. These I/O ports are then configured to communicate with peripheral devices. An I/O port can be an input port, an output port, or a bidirectional port. Some I/O ports are used for transmitting data, such as to and from the transmit and receive registers, respectively, of a serial interface device. Other I/O ports are used to control peripheral devices, such as the control registers of a disk controller.

This chapter describes the processor's I/O architecture. The topics discussed include:

- I/O port addressing.
- I/O instructions.
- I/O protection mechanism.

62.1 I/O Port Addressing

The processor allows I/O ports to be accessed in either of two ways:

- Through a separate I/O address space.
- Through memory-mapped I/O.

Accessing I/O ports through the I/O address space is handled through a set of I/O instructions and a special I/O protection mechanism. Accessing I/O ports through memory-mapped I/O is handled with the processors general-purpose move and string instructions, with protection provided through segmentation or paging. I/O ports can be mapped so that they appear in the I/O address space or the physical-memory address space (memory mapped I/O) or both.

One benefit of using the I/O address space is that writes to I/O ports are guaranteed to be completed before the next instruction in the instruction stream is executed. Thus, I/O writes to control system hardware cause the hardware to be set to its new state before any other instructions are executed. See "Ordering I/O" for more information on serializing of I/O operations.

62.2 I/O Port Hardware

From a hardware point of view, I/O addressing is handled through the processor's address lines. For Pentium Pro processors, a special memory-I/O transaction on the system bus indicates whether the address lines are being driven with a memory address or an I/O address; for Pentium and earlier Intel Architecture processors, the M/IO pin indicates a memory address (1) or an I/O address (0). When the separate I/O address space is selected, it is the responsibility of the hardware to decode the memory-I/O bus transaction to select I/O ports rather than memory.

Data is transmitted between the processor and an I/O device through the data lines.

62.3 I/O Address Space

The processor's I/O address space is separate and distinct from the physical-memory address space. The I/O address space consists of 2^{16} (64K) individually addressable 8-bit I/O ports, numbered 0 through FFFFH. I/O port addresses 0F8H through 0FFH are reserved. Do not assign I/O ports to these addresses. The result of an attempt to address beyond the I/O address space limit of FFFFH is implementation-specific; see the Developer's Manuals for specific processors for more details.

Any two consecutive 8-bit ports can be treated as a 16-bit port, and any four consecutive ports can be a 32-bit port. In this manner, the processor can transfer 8, 16, or 32 bits to or from a device in the I/O address space. Like words in memory, 16-bit ports should be aligned to even addresses (0, 2, 4, ...) so that all 16 bits can be transferred in a single bus cycle. Likewise, 32-bit ports should be aligned to addresses that are multiples of four (0, 4, 8, ...). The processor supports data transfers to unaligned ports, but there is a performance penalty because one or more extra bus cycle must be used.

The exact order of bus cycles used to access unaligned ports is undefined and is not guaranteed to remain the same in future Intel Architecture processors. If hardware or software requires that I/O ports be written to in a particular order, that order must be specified explicitly. For example, to load a word-length I/O port at address 2H and then another word port at 4H, two word-length writes must be used, rather than a single doubleword write at 2H.

Note that the processor does not mask parity errors for bus cycles to the I/O address space. Accessing I/O ports through the I/O address space is thus a possible source of parity errors.

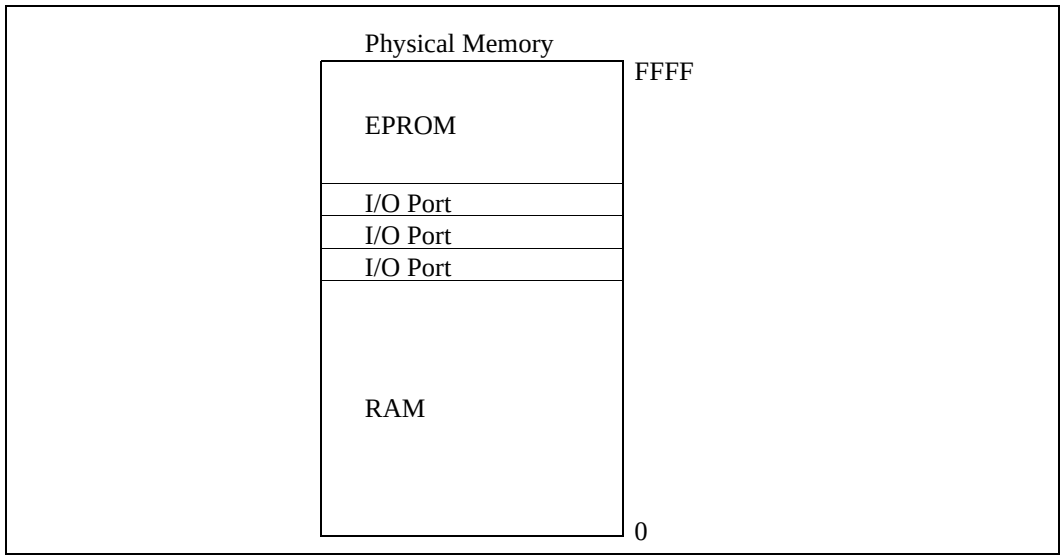
62.3.1 Memory-Mapped I/O

I/O devices that respond like memory components can be accessed through the processor's physical-memory address space (see Figure 62-1). When using memory-mapped I/O, any of the processor's instructions that reference memory can be used to access an I/O port located at a physical-memory address. For example, the MOV instruction can transfer data between any register and a memory-mapped I/O port. The AND, OR, and TEST instructions may be used to manipulate bits in the control and status registers of a memory-mapped peripheral devices.

When using memory-mapped I/O, caching of the address space mapped for I/O operations must be prevented. With the Pentium Pro processors, caching of I/O accesses can be prevented by using memory type range registers (MTRRs) to map the address space used for the memory-mapped I/O as uncacheable (UC). See Chapter 9, *Memory Cache Control*, in the *Intel Architecture Software Developer's Manual, Volume 3*, for a complete discussion of the MTRRs.

The Pentium and Intel486 processors do not support MTRRs. Instead, they provide the KEN# pin, which when held inactive (high) prevents caching of all addresses sent out on the system bus. To use this pin, external address decoding logic is required to block caching in specific address spaces.

Figure 62-1. Memory-Mapped I/O



All the Intel Architecture processors that have on-chip caches also provide the PCD (page-level cache disable) flag in page table and page directory entries. This flag allows caching to be disabled on a page-by-page basis. See “Page-Directory and Page-Table Entries” in Chapter 3 of in the *Intel Architecture Software Developer’s Manual, Volume 3.*

62.4 I/O Instructions

The processor’s I/O instructions provide access to I/O ports through the I/O address space. (These instructions cannot be used to access memory-mapped I/O ports.) There are two groups of I/O instructions:

- Those which transfer a single item (byte, word, or doubleword) between an I/O port and a general-purpose register.
- Those which transfer strings of items (strings of bytes, words, or doublewords) between an I/O port and memory.

The register I/O instructions IN (input from I/O port) and OUT (output to I/O port) move data between I/O ports and the EAX register (32-bit I/O), the AX register (16-bit I/O), or the AL (8-bit I/O) register. The address of the I/O port can be given with an immediate value or a value in the DX register.

The string I/O instructions INS (input string from I/O port) and OUTS (output string to I/O port) move data between an I/O port and a memory location. The address of the I/O port being accesses is given in the DX register; the source or destination memory address is given in the DS:ESI or ES:EDI register, respectively.

When used with one of the repeat prefixes (such as REP), the INS and OUTS instructions perform string (or block) input or output operations. The repeat prefix REP modifies the INS and OUTS instructions to transfer blocks of data between an I/O port and memory. Here, the ESI or EDI

register is incremented or decremented (according to the setting of the DF flag in the EFLAGS register) after each byte, word, or doubleword is transferred between the selected I/O port and memory.

See the IN, INS, OUT, and OUTS instructions for more information.

62.5 Protected-Mode I/O

When the processor is running in protected mode, the following protection mechanisms regulate access to I/O ports:

- When accessing I/O ports through the I/O address space, two protection devices control access:
  - The I/O privilege level (IOPL) field in the EFLAGS register.
  - The I/O permission bit map of a task state segment (TSS).
- When accessing memory-mapped I/O ports, the normal segmentation and paging protection and the MTRRs (in processors that support them) also affect access to I/O ports. See Chapter 4, *Protection*, and Chapter 9, *Memory Cache Control*, in the *Intel Architecture Software Developer's Manual, Volume 3*, for a complete discussion of memory protection.

The following sections describe the protection mechanisms available when accessing I/O ports in the I/O address space with the I/O instructions.

62.5.1 I/O Privilege Level

In systems where I/O protection is used, the IOPL field in the EFLAGS register controls access to the I/O address space by restricting use of selected instructions. This protection mechanism permits the operating system or executive to set the privilege level needed to perform I/O. In a typical protection ring model, access to the I/O address space is restricted to privilege levels 0 and 1. Here, kernel and the device drivers are allowed to perform I/O, while less privileged device drivers and application programs are denied access to the I/O address space. Application programs must then make calls to the operating system to perform I/O.

The following instructions can be executed only if the current privilege level (CPL) of the program or task currently executing is less than or equal to the IOPL: IN, INS, OUT, OUTS, CLI (clear interrupt-enable flag), and STI (set interrupt-enable flag). These instructions are called **I/O sensitive** instructions, because they are sensitive to the IOPL field. Any attempt by a less privileged program or task to use an I/O sensitive instruction results in a general-protection exception (#GP) being signaled. Because each task has its own copy of the EFLAGS register, each task can have a different IOPL.

The I/O permission bit map in the TSS can be used to modify the effect of the IOPL on I/O sensitive instructions, allowing access to some I/O ports by less privileged programs or tasks (see "I/O Permission Bit Map").

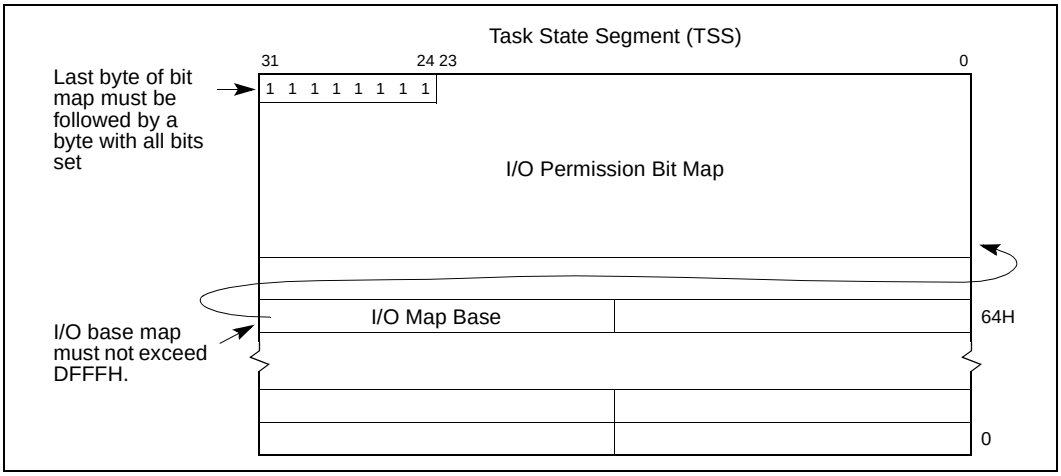
A program or task can change its IOPL only with the POPF and IRET instructions; however, such changes are privileged. No procedure may change the current IOPL unless it is running at privilege level 0. An attempt by a less privileged procedure to change the IOPL does not result in an exception; the IOPL simply remains unchanged.

The POPF instruction also may be used to change the state of the IF flag (as can the CLI and STI instructions); however, the POPF instruction in this case is also I/O sensitive. A procedure may use the POPF instruction to change the setting of the IF flag only if the CPL is less than or equal to the current IOPL. An attempt by a less privileged procedure to change the IF flag does not result in an exception; the IF flag simply remains unchanged.

62.5.2 I/O Permission Bit Map

The I/O permission bit map is a device for permitting limited access to I/O ports by less privileged programs or tasks and for tasks operating in virtual-8086 mode. The I/O permission bit map is located in the TSS (see Figure 62-2) for the currently running task or program. The address of the first byte of the I/O permission bit map is given in the I/O map base address field of the TSS. The size of the I/O permission bit map and its location in the TSS are variable.

Figure 62-2. I/O Permission Bit Map



Because each task has its own TSS, each task has its own I/O permission bit map. Access to individual I/O ports can thus be granted to individual tasks.

If in protected mode and the CPL is less than or equal to the current IOPL, the processor allows all I/O operations to proceed. If the CPL is greater than the IOPL or if the processor is operating in virtual-8086 mode, the processor checks the I/O permission bit map to determine if access to a particular I/O port is allowed. Each bit in the map corresponds to an I/O port byte address. For example, the control bit for I/O port address 29H in the I/O address space is found at bit position 1 of the sixth byte in the bit map. Before granting I/O access, the processor tests all the bits corresponding to the I/O port being addressed. For a doubleword access, for example, the processors tests the four bits corresponding to the four adjacent 8-bit port addresses. If any tested bit is set, a general-protection exception (#GP) is signaled. If all tested bits are clear, the I/O operation is allows to proceed.

Because I/O port addresses are not necessarily aligned to word and doubleword boundaries, the processor read two bytes from the I/O permission bit map for every access to an I/O port. To prevent exceptions from being generated when the ports with the highest addresses are accessed, an extra byte needs to included in the TSS immediately after the table. This byte must have all of its bits set, and it must be within the segment limit.

It is not necessary for the I/O permission bit map to represent all the I/O addresses. I/O addresses not spanned by the map are treated as if they had set bits in the map. For example, if the TSS segment limit is 10 bytes past the bit-map base address, the map has 11 bytes and the first 80 I/O ports are mapped. Higher addresses in the I/O address space generate exceptions.

If the I/O bit map base address is greater than or equal to the TSS segment limit, there is no I/O permission map, and all I/O instructions generate exceptions when the CPL is greater than the current IOPL. The I/O bit map base address must be less than or equal to DFFFH.

62.6 Ordering I/O

When controlling I/O devices it is often important that memory and I/O operations be carried out in precisely the order programmed. For example, a program may write a command to an I/O port, then read the status of the I/O device from another I/O port. It is important that the status returned be the status of the device **after** it receives the command, not **before**.

When using memory-mapped I/O, caution should be taken to avoid situations in which the programmed order is not preserved by the processor. To optimize performance, the processor allows cacheable memory reads to be reordered ahead of buffered writes in most situations. Internally, processor reads (cache hits) can be reordered around buffered writes. When using memory-mapped I/O, therefore, is possible that an I/O read might be performed before the memory write of a previous instruction. The recommended method of enforcing program ordering of memory-mapped I/O accesses with the Pentium Pro processor is to use the MTRRs to make the memory mapped I/O address space uncacheable; for the Pentium and Intel486 processors, either the #KEN pin or the PCD flags can be used for this purpose (see “Memory-Mapped I/O”). When the target of a read or write is in an uncacheable region of memory, memory reordering does not occur externally at the processor’s pins (that is, reads and writes appear in-order). Designating a memory mapped I/O region of the address space as uncacheable insures that reads and writes of I/O devices are carried out in program order. See Chapter 9, *Memory Cache Control*, in the *Intel Architecture Software Developer’s Manual, Volume 3*, for more information on using MTRRs.

Another method of enforcing program order is to insert one of the serializing instructions, such as the CPUID instruction, between operations. See Chapter 7, *Multiple Processor Management*, in the *Intel Architecture Software Developer’s Manual, Volume 3*, for more information on serialization of instructions.

It should be noted that the chip set being used to support the processor (bus controller, memory controller, and/or I/O controller) may post writes to uncacheable memory which can lead to out-of-order execution of memory accesses. In situations where out-of-order processing of memory accesses by the chip set can potentially cause faulty memory-mapped I/O processing, code must be written to force synchronization and ordering of I/O operations. Serializing instructions can often be used for this purpose.

When the I/O address space is used instead of memory-mapped I/O, the situation is different in two respects:

- The processor never buffers I/O writes. Therefore, strict ordering of I/O operations is enforced by the processor. (As with memory-mapped I/O, it is possible for a chip set to post writes in certain I/O ranges.)
- The processor synchronizes I/O instruction execution with external bus activity (see Table 62-1).

Table 62-1. I/O Instruction Serialization

Instruction Being Executed	Processor Delays Execution of …		Until Completion of …	
	Current Instruction?	Next Instruction?	Pending Stores?	Current Store?
IN	Yes		Yes	
INS	Yes		Yes	
REP INS	Yes		Yes	
OUT		Yes	Yes	Yes
OUTS		Yes	Yes	Yes
REP OUTS		Yes	Yes	Yes